Hadoop 雲端運算平台效能模式之評估與改善

資工三甲 409261055 柯承佑

資工三甲 409261108 陳新義

資工三甲 409261548 陳祈叡

資工三甲 409261641 張庭豪

摘要

- 現今資訊發達的年代，「巨量資料」是相當重要的一部分。
- 雲端平台的特性之一就是用來處理這些巨量資料，以解決在大量資料下，運算資源或是儲存空間不足等問題。
- 雲端平台中， **Hadoop 平台**的效能議題開始被做研究及討論。

訴諸問題

欲提高 Hadoop 效能

- **透過更改 Hadoop 原始碼**

  → 非常耗時。

- **直接調整 Hadoop 中配置參數**

  → 當中有多種組合與類型的參數，要直接找出適當的組合來做調整也是相當複雜，且也可能造成其他副作用。

解決方式

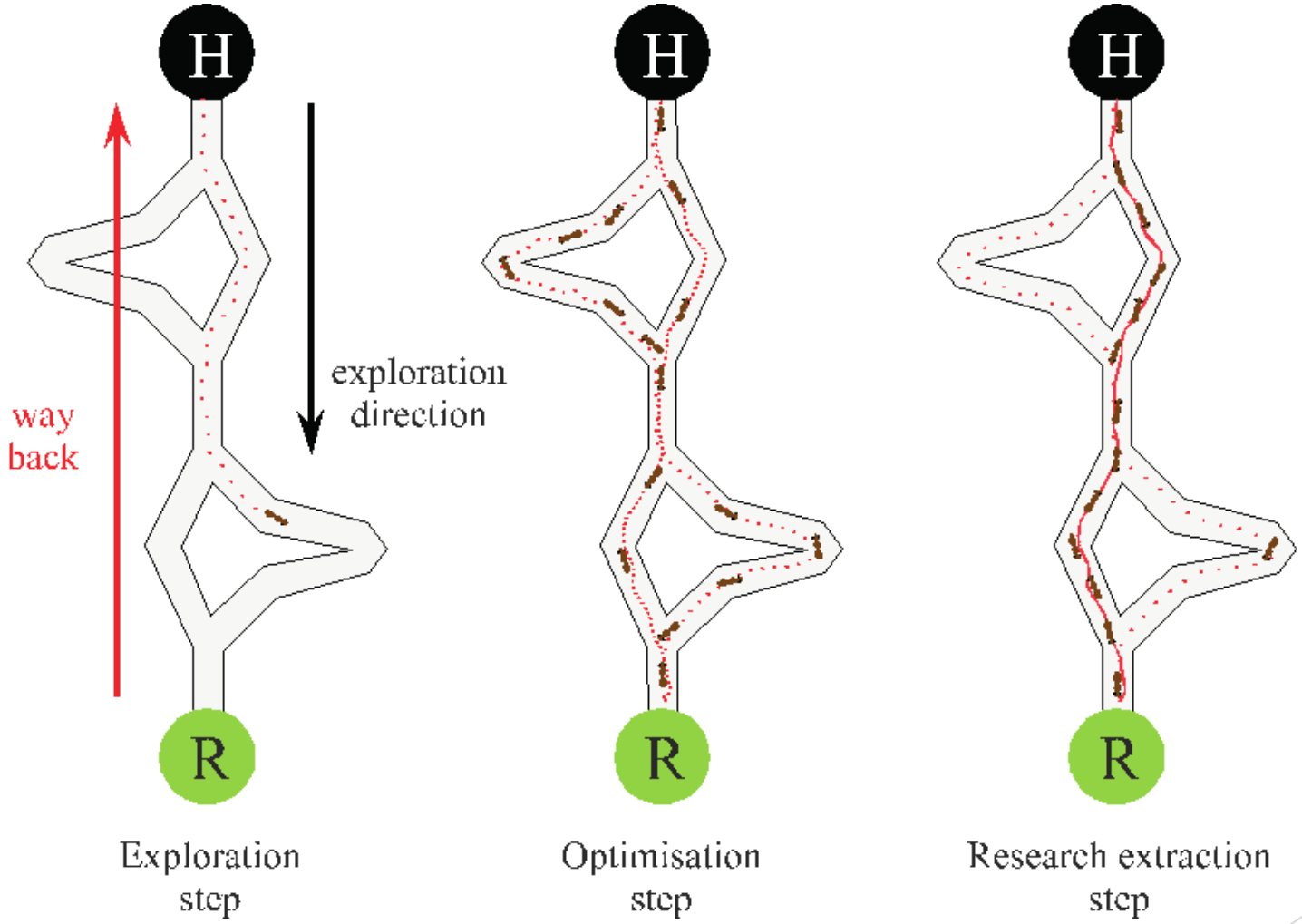
- 建立一個跨 IaaS、PaaS、SaaS 三層的模型(跨層式參數優化模型，CPOM)，並將「**螞蟻演算法**」加入模型中，以找出較加參數組合，提高 Hadoop 的效能。
- 實驗部分則是透過 Hadoop 實際操作來驗證 CPOM 的可行性。

螞蟻演算法

- 螞蟻覓食過程中，會在行走過的路線留下費洛蒙，同時也作為其他螞蟻走該路線的依據。
- 費洛蒙會隨時間慢慢蒸發，路線越短，殘留費洛蒙濃度越高。
- 螞蟻遇到多條岔路時，會往費洛蒙濃度越高的路線走。
- 一段時間後，所有螞蟻沿著同一條路線走。

  →該路線即為「最佳路徑」。

螞蟻演算法

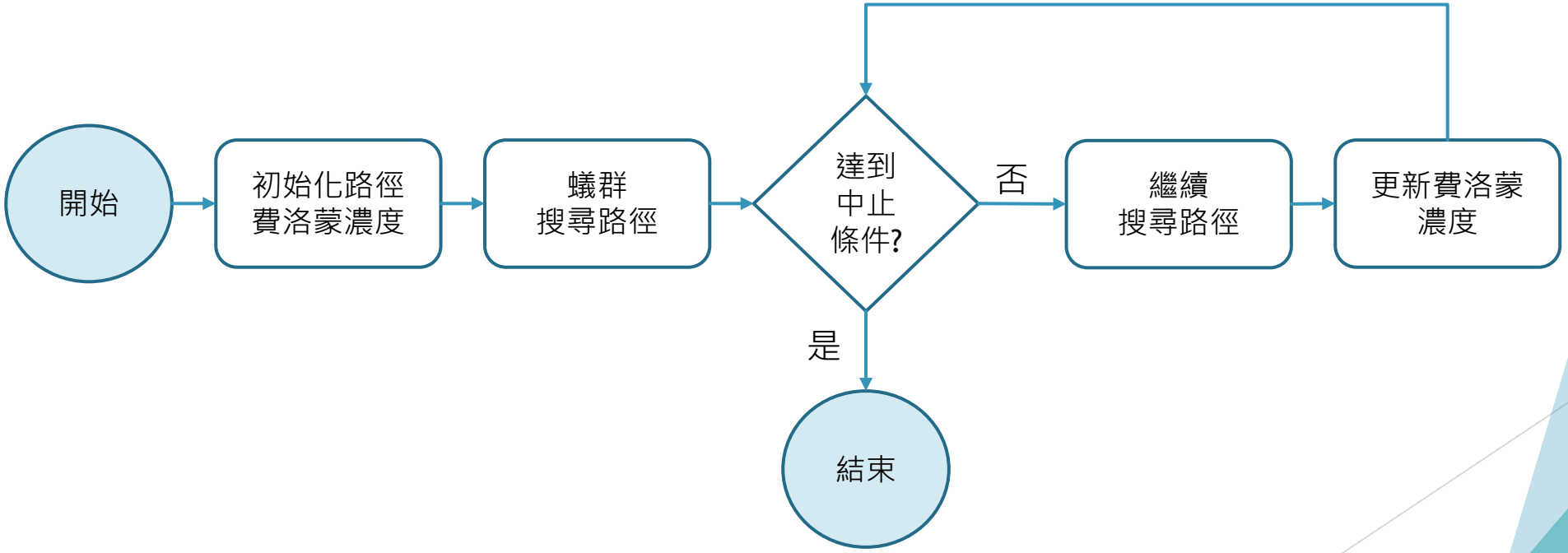

H
R
way
back
exploration
direction
Exploration
step
H
R
Optimisation
step
H
R
Research extraction
step

螞蟻演算法

- 螞蟻演算法:利用**人工螞蟻**來模擬螞蟻找尋最佳路徑行為。
- **螞蟻族群系統**（Ant Colony System，ACS)，螞蟻演算法模型中應用較為廣泛的一種。

螞蟻演算法

- 螞蟻演算法流程圖



螞蟻演算法

ACS 流程

- 選擇鄰居節點
- 區域費洛蒙更新
- 全域費洛蒙更新

螞蟻演算法

選擇鄰居節點

- 初始化費洛蒙濃度 τ_{ij} =C。
- 依**狀態轉移法則**(State Transition Rule)尋找下一鄰居節點。

螞蟻演算法

狀態轉移法則

- 螞蟻 k 在節點 i 選擇某鄰居 j 的機率為

$$P_{ij}^{k}(t)=\frac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}(t)]^{\beta}}{\sum_{l\in allowed_k}[\tau_{il}(t)]^{\alpha}\cdot[\eta_{il}(t)]^{\beta}}\quad if\quad j\in allowed_k$$

$\tau_{ij}(t)$:時間t 時，鏈結（i, j）上的費洛蒙濃度。

$\eta_{ij}(t)=1/dij$:節點i與鄰居 j之間的能見度。

d:距離。 α :費洛蒙濃度參數。 B:絕對距離參數。

螞蟻演算法

區域費洛蒙更新

- 當螞蟻經過一段路徑，即減少該路徑費洛蒙量。

  →避免螞蟻走同一段路。

- 費洛蒙更新公式:

$$\tau_{ij}(t) = (1-\theta) \times \tau_{ij}(t) + \theta \times \Delta\tau_0$$

$0 < \theta < 1$, τ_0為費洛蒙的初始值

螞蟻演算法

全域費洛蒙更新

- 每回合結束後只在最佳路徑增加費洛蒙。

  →代表該路徑為當回合最佳解。

- 費洛蒙更新公式:

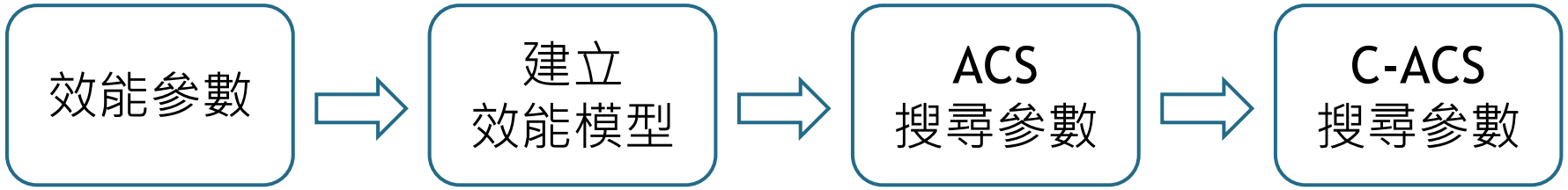
$$\tau_{ij}(t+1) = (1-\rho) \times \tau_{ij}(\tau) + \rho \times \Delta\tau_{ij}(t)$$

$$\Delta\tau_{ij}(t) = 1/f^{gb}(t) \text{ if 鏈結}(i, j) \in \text{最佳路徑}$$

跨層式參數優化模型

跨層式參數優化模型

(Cross-Layer Parameters Optimization Model, CPOM)



▲CPOM 架構

1.CPOM 效能參數

- Hadoop 中約有200多種可調整參數。
- 經研究後，將參數縮減為約20多個重要**效能參數**。
- 利用這些**效能參數**加上**跨層效能參數**來做出模型。

1.CPOM 效能參數

SaaS 層

- 運行 **MapReduce** 處。
- 效能參數分為以下三種
  - Job 整體效能參數
  - MapTask 效能參數
  - ReduceTask 效能參數

1.CPOM 效能參數

- Job 整體效能參數

參數簡稱	參數名稱	參數介紹	建議值
nM	mapred.map.tasks	執行的 Map 任務數量	CPU/2~ CPU*2
nR	mapred.reduce.tasks	執行的 Reduce 任務數量	CPU/2~ CPU*2
mJVM	mapred.child.java.opts	JVM 記憶體容量	80% × Mem
MapCompress	mapred.compress.map.output	Map 輸出是否要 壓縮	True
MapCodec	mapred.map.output.compression.codec	壓縮格式	LZO

1.CPOM 效能參數

MapTask 效能參數

參數編號	參數名稱	參數介紹
1	io.sort.mb	緩衝區大小
2	io.sort.factor	合併檔案數目
3	io.sort.record.percent	緩衝區中 Record 的比例
4	io.sort.spill.percent	溢出到硬碟門檻值
5	min.num.spill.for.combine	最少 combine 數量

ReduceTask 效能參數

參數編號	參數名稱	參數介紹
1	mapred.job.shuffle.input.buffer.percent	Shuffle JVM heap size
2	mapred.job.shuffle.merge.percent	合併門檻值
3	mapred.job.reduce.input.buffer.percent	Reduce 儲存百分比
4	mapred.reduce.slowstart.completed.maps	何時啟動 Reduce
5	mapred.inmem.merge.threshold	Map 讀取資料門檻值

1.CPOM 效能參數

PaaS 層

- 搜尋與告知有哪些 VM 是可使用的 。
- **PaaS 層效能參數**

參數簡稱	參數名稱	參數介紹
nSlave	Slave	工作節點數量
nMapSlot	mapred.tasktracker.map.tasks.maximum	最大 MapSlot 數量
nReduceSlot	mapred.tasktracker.reduce.tasks.maximum	最大 ReduceSlot 數量

1.CPOM 效能參數

IaaS 層

- 管理 Host 與 VM 的執行能力，並找出適合的硬體資源。

- **IaaS 層效能參數**

參數簡稱	參數介紹
nHostr	Host 的數量
hCPU	Host 的 CPU 能力
hMemory	Host 的記憶體容量
hIO	Host 的 IO 能力

(Host 參數)

參數簡稱	參數介紹
nVM	Vm 的數量
vCPU	Vm 的 CPU 能力
vMemory	Vm 的記憶體容量
vIO	Vm 的 IO 能力

(VM 參數)

2.建立效能模型

- 在找出效能參數與 Hadoop 結合後，得出效能模型公式如下。

$$T = T_{Setup} + T_{Map} + T_{Reduce} + T_{Cleanup}$$

3.ACS 搜尋參數

- 以螞蟻演算法找出 SaaS 層中的最佳參數值組合。

SaaS 層優化

- 執行 Hadoop 中 Job 的調教。
- 目標函式: $\boldsymbol{Minimize\ P_j^s = T_j - T_{j-1}}$

  →用於計算減少執行時間的效能成本。

4.C-ACS

- 將 **PaaS 與 IaaS 層**的影響也加入考量。
- 從 IaaS 層優化 → PaaS 層優化 → SaaS 層優化，構成 CPOM。

4.C-ACS

PaaS 層優化

- 將有空閒的 VM 暫時加入叢集內。
- 目標函式: $\boldsymbol{Minimize\ P_j^p = S_j^{total} - S_j^{idle}}$

  →用於計算實體機的忙碌資源。

4.C-ACS

IaaS 層優化

- 找出較加的虛擬機與實體機搭配方式，以產生環境最佳組合。
- 目標函式: $\boldsymbol{Minimize\ P_j^i = T_j - T_{j-1}}$

  →計算執行時間的效能改善成本。

4.C-ACS

跨層目標函式 F(x)

- ACS 透過搜尋方式，得出F(x)，達成跨層優化目的。
- 跨層目標函式給予更多的最佳化參數選擇。

$$\min F(x)=P_j^{\,i}+P_j^{\,p}+P_j^{\,s}$$

4.C-ACS

CPOM 運作架構

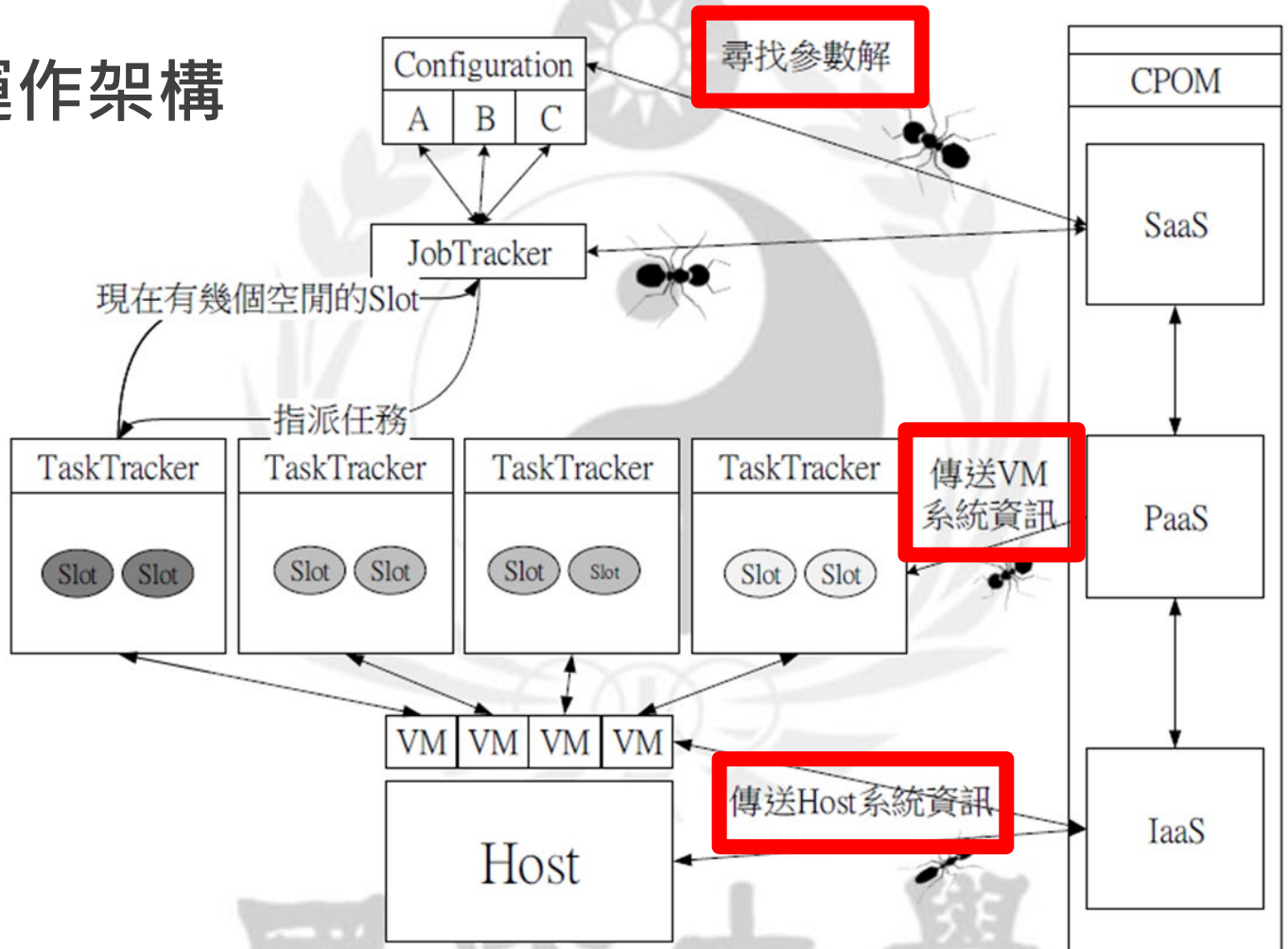


圖 3- 5 CPOM 運作架構

實驗

- 建立了一個虛擬 Hadoop 叢集
- 硬體與軟體配置:

Server	CPU	INTEL Xeon E3-1230V2 quad-core at 3.3GGHz
	Memory	32GB
	Disk	1000GB
Software	OS	Linux CentOS6.3_64bit
	Hadoop	Hadoop 1.0.4
	Java	Sun JDK 6u21

實驗環境上所配置的 6 個節點的環境規格:

節點名稱	Vcpu	vMemory
Master	2	4096MB
Slave1	2	4096MB
Slave2	2	4096MB
Slave3	2	4096MB
Slave4	2	4096MB
Slave5	2	4096MB

實驗: 找出參數最佳化的組合

取出 4 種 效能參數，每種效能參數給定 4 個數值，總共 16 個數值。透過執行螞蟻演算法，根據執行時間找到適合這個負載的參數設定。

參數種類	編號	參數值	執行時間	參數種類	編號	參數值	執行時間
io.sort.mb	A1	100	144	io.sort.record.percent	C1	0.05	144
	A2	150	138		C2	0.1	127
	A3	200	124		C3	0.15	156
	A4	250	131		C4	0.2	172
io.sort.factor	B1	10	144	mapred.reduce.tasks	D1	1	144
	B2	50	141		D2	3	126
	B3	75	150		D3	5	135
	B4	100	159		D4	10	126

WordCount 執行時間

實驗: 找出參數最佳化的組合

參數種類	編號	參數值	執行時間	參數種類	編號	參數值	執行時間
io.sort.mb	A1	100	100	io.sort.record.percent	C1	0.05	100
	A2	150	98		C2	0.1	93
	A3	200	89		C3	0.15	85
	A4	250	112		C4	0.2	92
io.sort.factor	B1	10	100	mapred.reduce.tasks	D1	1	100
	B2	50	99		D2	3	70
	B3	75	88		D3	5	79
	B4	100	81		D4	10	82

Terasort 執行時間

實驗: 找出參數最佳化的組合

與未調整的時間進行比較，發現最佳化後的參數的確可以改善效能問題。

WordCount		Terasort	
參數種類	搜尋最佳解	參數種類	搜尋最佳解
io.sort.mb	200	io.sort.mb	200
io.sort.factor	50	io.sort.factor	100
io.sort.record.percent	0.1	io.sort.record.percent	0.15
mapred.reduce.tasks	3	mapred.reduce.tasks	3

CPR 最佳化參數

總結

- 從各層找出效能參數，並做成效能模型。
- 最終經由螞蟻演算法與各層目標函式，達成單層與跨層優化目的。
- 從實驗結果得知，CPOM 的確可改善 Hadoop 效能。

資料來源

- Hadoop 雲端運算平台效能模式之評估與改善

報告結束~